

Fig. 1

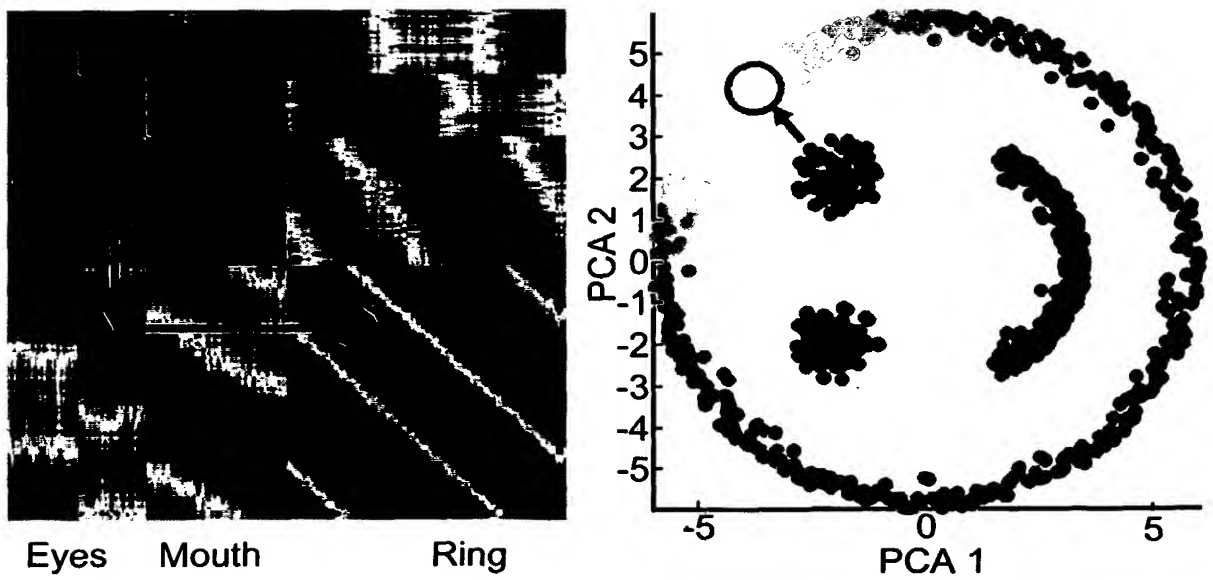
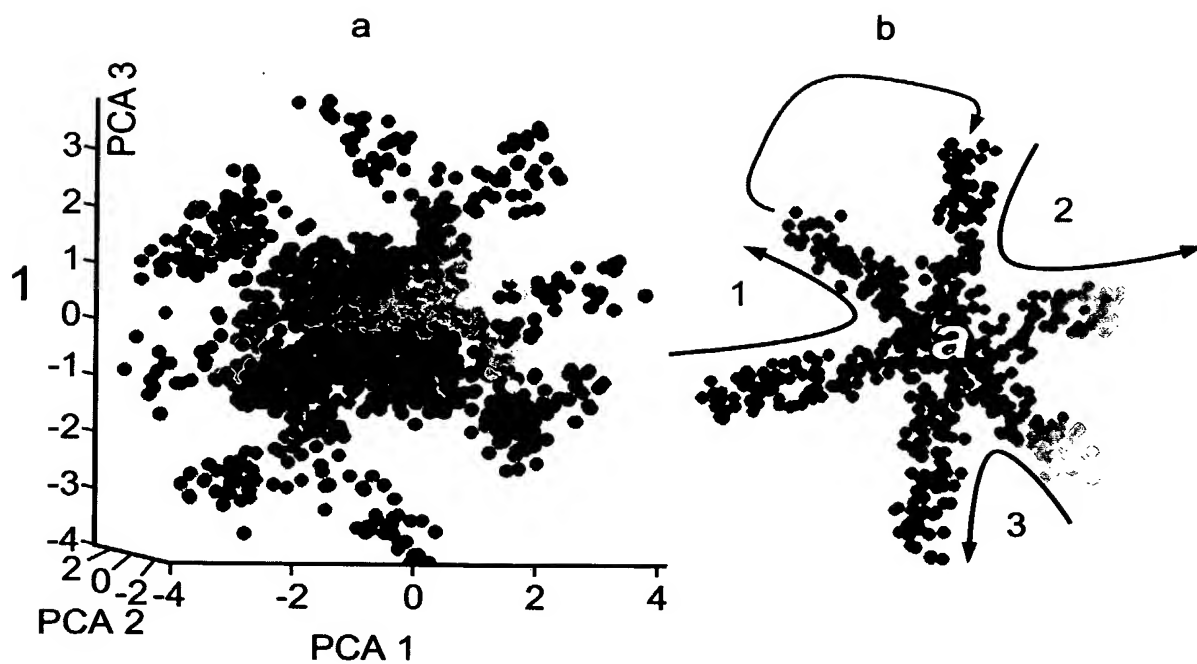
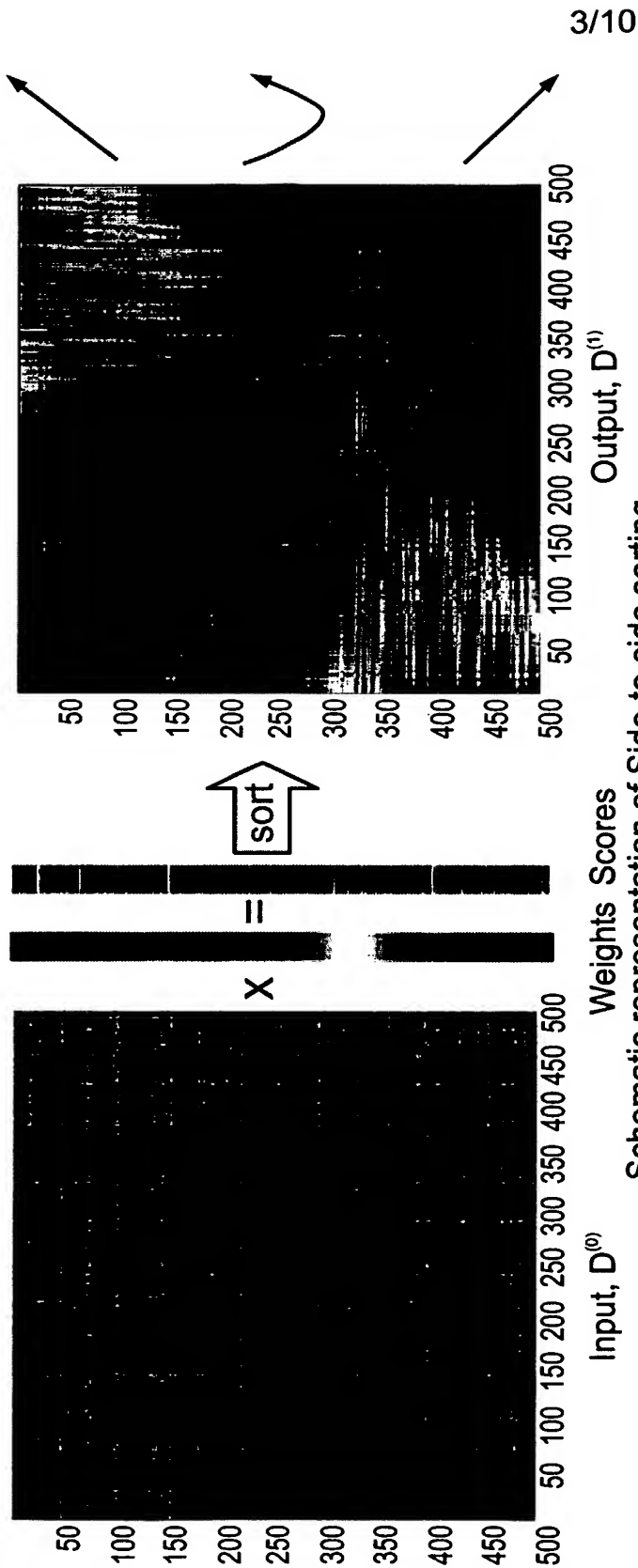


Fig. 2



2

Fig. 3



Schematic representation of Side-to-side sorting

The initial unsorted distance matrix on the left is multiplied by a weight vector, resulting in a vector of scores. The weight vector's components increase linearly from -1 to 1. The points are then sorted according to their scores, generating a new permutation of the distance matrix, as shown on the right. This process is iterated until convergence, and the final outcome is shown at the right. By viewing the sorted matrix on the right it is readily apparent that the overall trend of the values in the upper rows is ascending, while the bottom rows have descending values, and intermediate rows have in-between values.

Fig. 4

Side-by-side

1. Define the weight vector $W_j = \frac{2j - N - 1}{N - 1}$
2. Calculate the scores vector $S_i^{(t)} = \sum_j D_{ij}^{(t)} W_j$
3. Sort the scores $\{k\} = \text{index sort}(\{S_i\})$
4. Reorder the distance matrix $D^{(t+1)} = D^{(t)}(\{k\}, \{k\})$
5. Repeat steps 2-4 until $D^{(t+1)}$ is equal to $D^{(t)}$

Fig. 5

Results of Side-by-side Algorithm

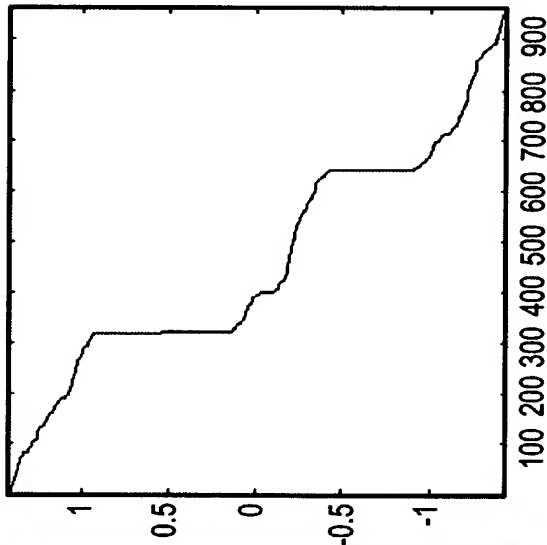
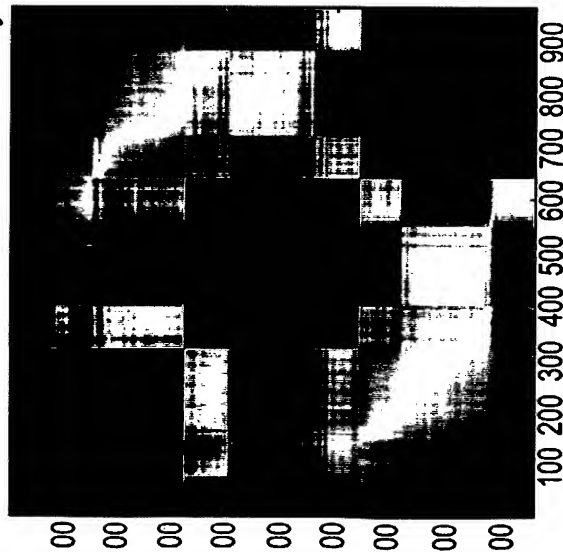
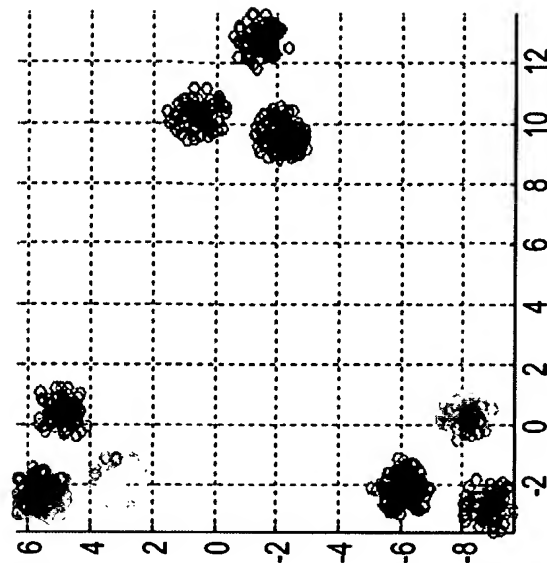


Fig. 6



Neighborhood Algorithm

1. Define the weight matrix $W_{ij} = e^{\frac{-(i-j)^2}{\epsilon \cdot N}} / \sum_k e^{\frac{-(k-j)^2}{\epsilon \cdot N}}$
2. Calculate the mismatch matrix $M_{ij}^{(t)} = \sum_k D_{ik}^{(t)} W_{kj}$
3. Extract score vector $S_i = \arg \min_j (M_{ij})$
4. Sort the scores $\{k\} = \text{index sort}(\{S_i\})$
5. Reorder the distance matrix $D^{(t+1)} = D^{(t)}(\{k\}, \{k\})$
6. Repeat steps 1-5 while adjusting ϵ

Fig. 7

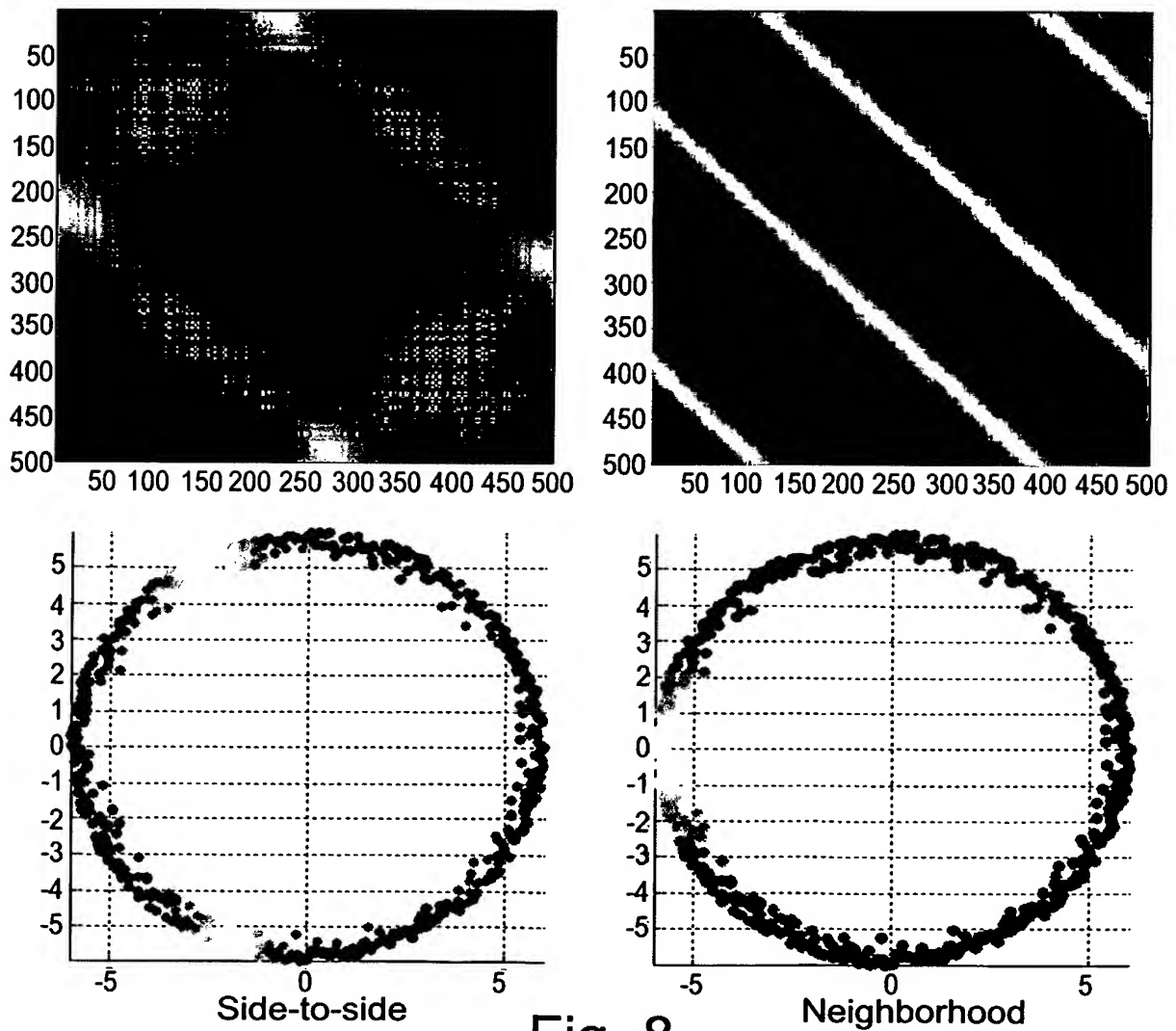


Fig. 8

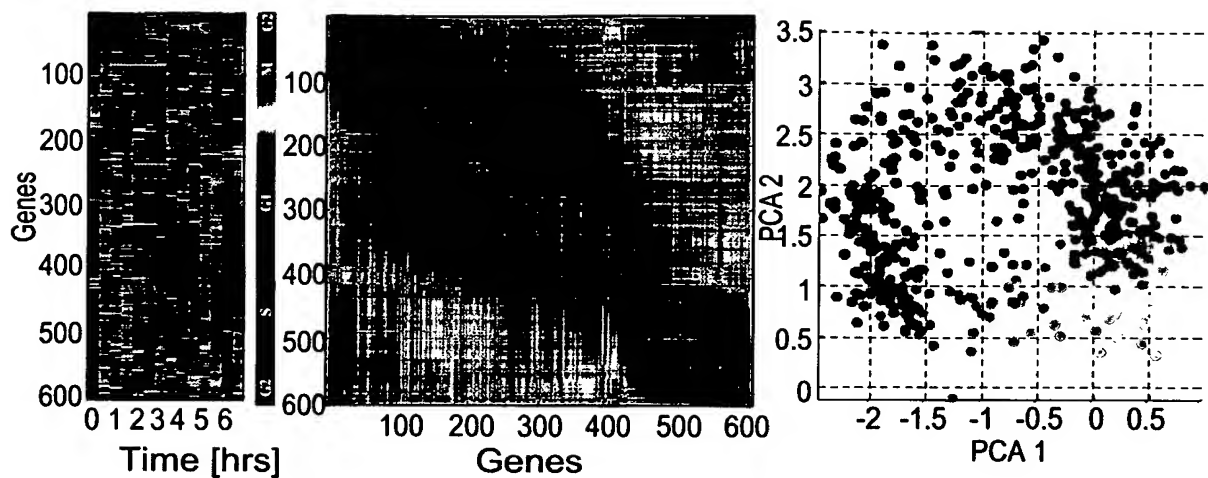


Fig. 9

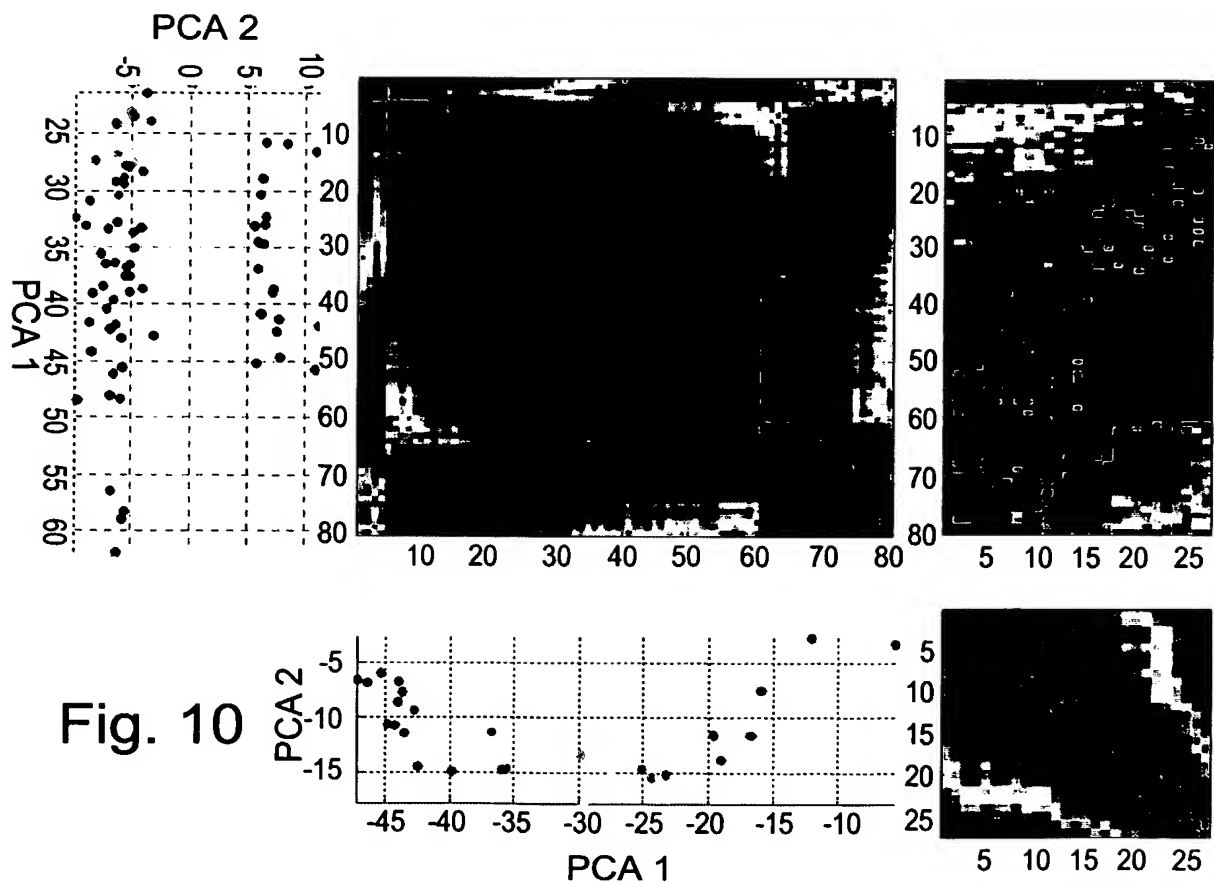


Fig. 10

7/10

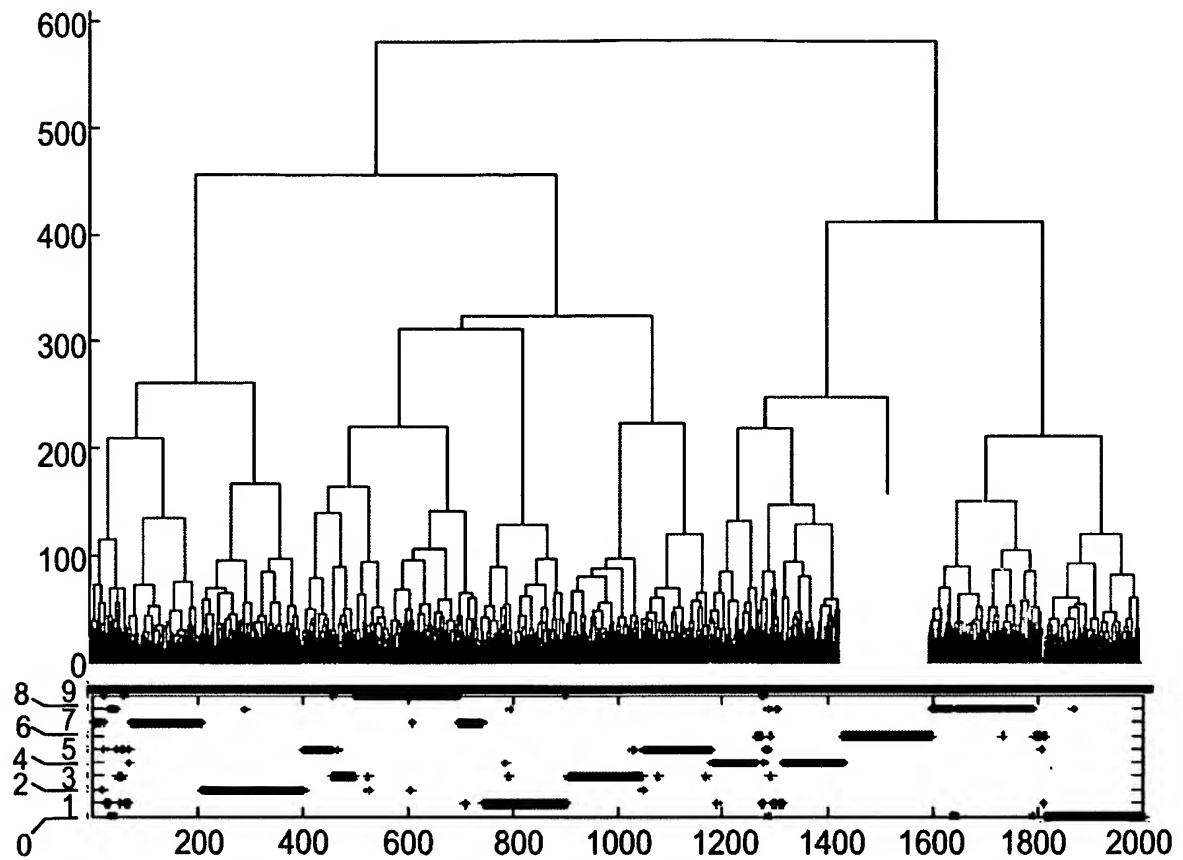


Fig. 11a

8/10

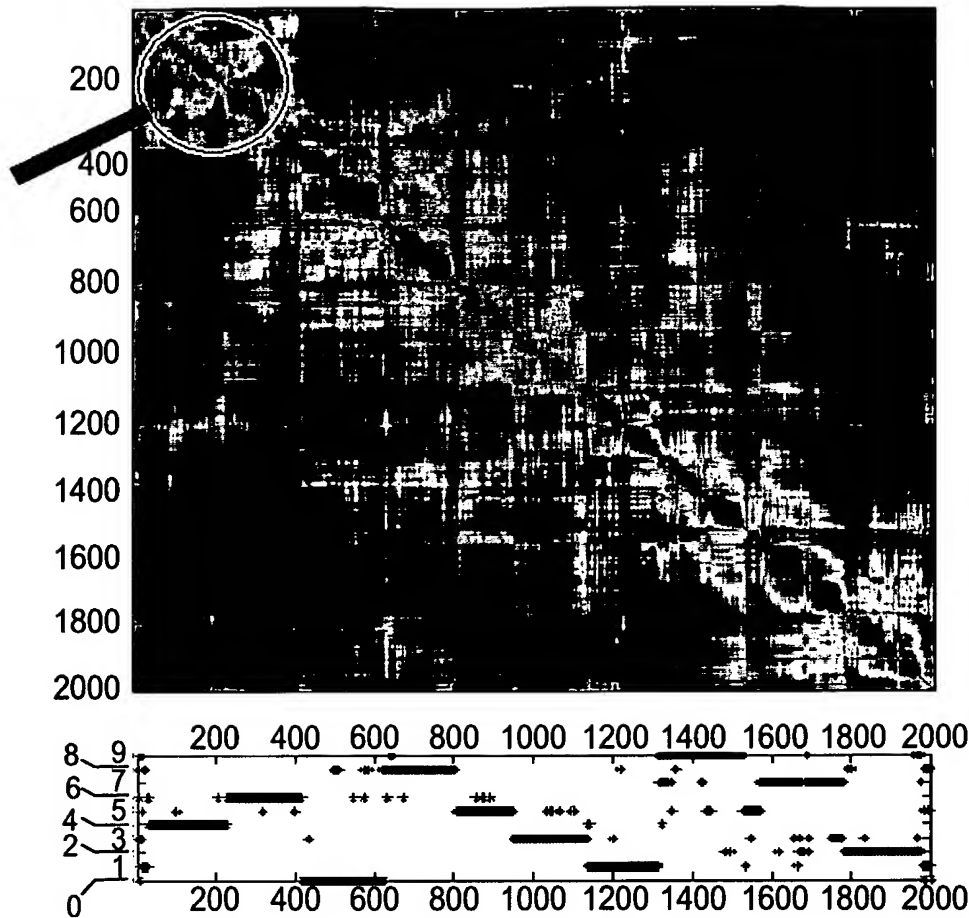


Fig. 11b

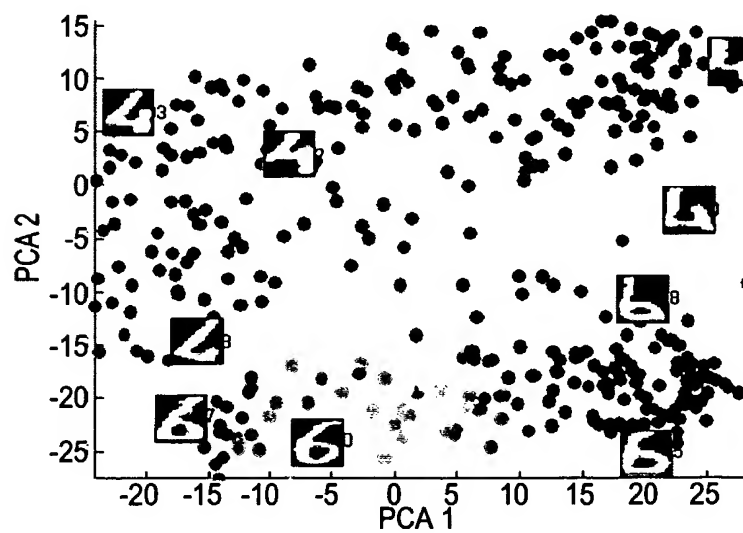


Fig. 11c

9/10

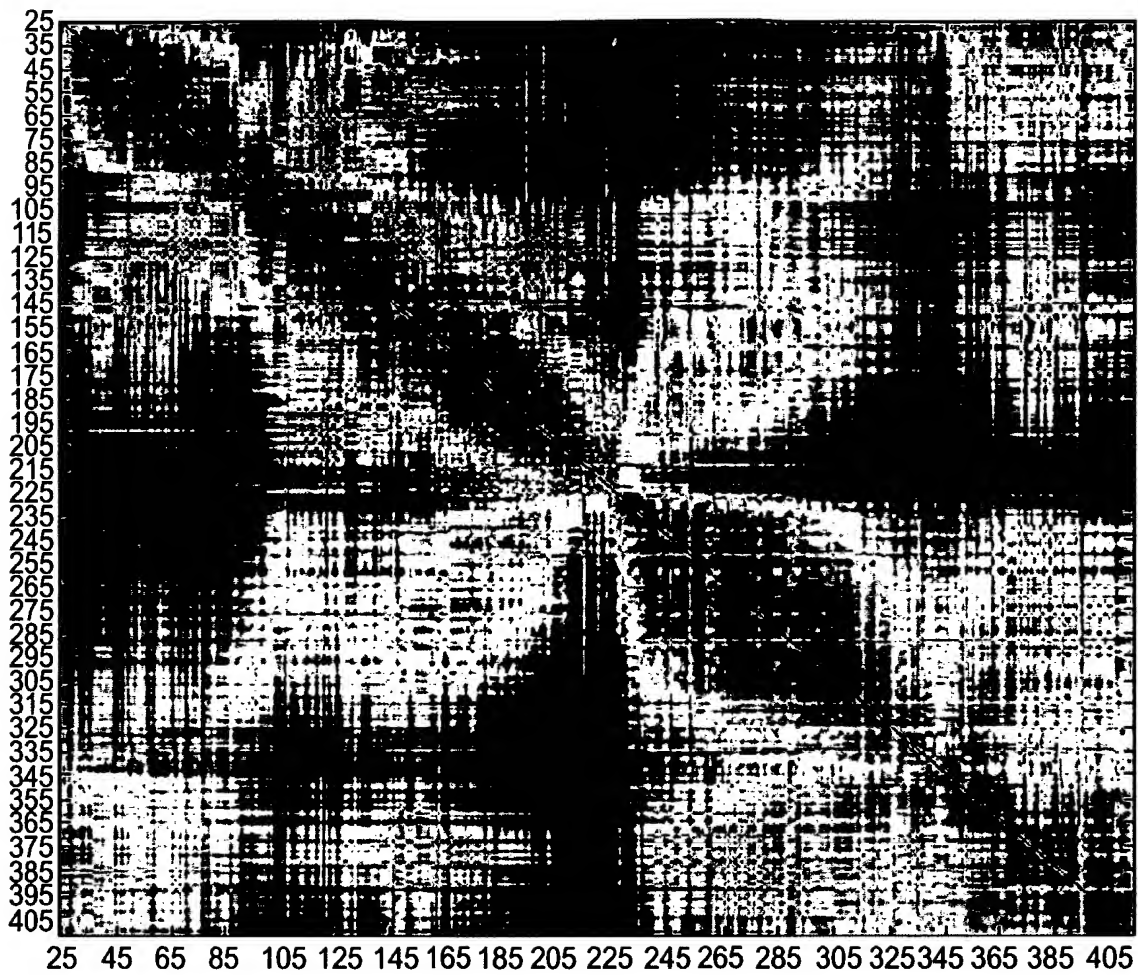


Fig. 11d



Fig. 11e

